Inner Classes

- A class within another class.
- Inner classes have access to other members of the enclosing class even private!
- An instance of InnerClass can exist only within an instance of OuterClass

Why InnerClasses?
- It is a way of logically grouping classes that are only used in one place.
- It increases encapsulation.
- Nested classes can lead to more readable and maintainable code
Inner Classes

Two types of inner classes:

- **Local Classes**
  - Can be defined inside any block, method body, for loop, if clause.
  - Can access method of the outer class.
  - Can access local variables if they are defined final.

- **Anonymous classes**
  - Enable to declare and instantiate a class at the same time.
  - They do not have a name.
  - Use them if you need to use your local class only once.
  - Must always implement an interface or extend an abstract class.
  - **Syntax:** `new interface-or-class-name() { class-body }`
  - Cannot have a constructor
  - Can access any variables visible to the block within which the anonymous class is declared, including final local variables.
  - Can access methods of the class that contains it.
class outerClass{
    int counter = 0;
    public outerClass(){
        InnerClass myObj1 = new InnerClass("ONE");
        counter++;
        InnerClass myObj2 = new InnerClass("TWO");
        System.out.println(myObj2);
        System.out.println(myObj1);
    }
}

class InnerClass{
    String name;
    int index;
    public InnerClass(String param){
        name = param;
        index = counter;
        counter++;
    }
    public String toString(){
        return "InnerClass: Name="+name+" Index="+index;
    }
}
interface myInterface{
    public void myMethod();
}

class outerClass{
    String msg = "Original";

    public void outerMethod(){
        myInterface obj = new myInterface(){
            public void myMethod(){
                msg = "Changed";
            }
        };
    }

    public String toString(){
        return "outerClass: msg="+msg;
    }
}
interface myInterface{
    public void myMethod();
}

class outerClass{
    String msg = "Original";
    public void outerMethod(){
        myInterface obj = new myInterface(){
            public void myMethod(){
                msg = "Changed";
            }
        };
    }
    public String toString(){
        return "outerClass: msg="+msg;
    }
    public void callMyMethod(){
        obj.myMethod();
    }
}

What does this code print:
outerClass obj = new outerClass();
obj.outerMethod();
obj.callMyMethod();
System.out.println(obj);
interface myInterface{
    public void myMethod();
}

class outerClass{
    String msg = "Original";
    myInterface obj;
    public void outerMethod(){
        obj = new myInterface(){
            public void myMethod(){
                msg = "Changed";
            }
        };
    }
    public String toString(){
        return "outerClass: msg=\"+msg;"
    }
    public void callMyMethod(){
        obj.myMethod();
    }
}
BoardCell is an inner class of ConnectFour so it can access all methods and instance variables of ConnectFour.

For the "New Game" button you can use an anonymous class that implements ActionListener.  
- What method must you implement?  
- When does this method gets called?  
- What do you need to do in that method?
1. To what type of object will you add an instance of a PlayListener object (as a MouseListener)?
2. If your board has width w and height h, how many BoardCell objects do you need to create? To which component will you add these BoardCell objects?
3. Why is the JLabel status an instance variable (as opposed to just a local variable in the constructor)?
4. Where is the information about what contents are stored in each cell located? What method must the BoardCell call in its paintComponent method to determine what color to paint the "checker"?
5. Which method will determine when the game is over (by calling methods on the ConnectFourBoard object theBoard)? Which method will detect illegal moves (again by calling methods on the ConnectFourBoard object theBoard)?
6. Will you need to create a separate listener to handle clicks on the New Game button, or will you use another instance of the PlayListener class?
7. How do you run the game?
PSA6 Common Errors

- Unknown variables or objects
  - Make sure you have your class inside the class from where you are trying to access it.
  - Make sure you have your curly brackets correct.
- Status label doesn't get updated
  - Use the provided status JLabel object and change the text with setText().
- Create BoardCell objects and try to add the PlayListener to them (Cannot access PlayListener)
  - Add the PlayListener inside BoardCell constructor.
- Checker doesn't get painted
  - Use getContents() to get the cell content.
  - How to use getContents().
- What to do in MakeMove()?
- Using PlayListener vs ActionListener
  - PlayListener extends MouseListener (use for mouse events)
  - ActionListener handles button clicks (complete clicks, space bar, etc)
Questions?

Office Hour: Today 4:30pm B260A